```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
```

```
import nltk
import re
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

Step1: Importing and downloading required libraries. Step2: Creating document.

```
text = "Gayatri is sitting beside me, doing practical A7 of DSBDA. Harish is sitting far away which is very sad."
from nltk.tokenize import sent_tokenize
var = sent_tokenize(text)
print(var)
```

```
['Gayatri is sitting beside me, doing practical A7 of DSBDA.', 'Harish is sitting far away which is very sad.']
```

```
from nltk.tokenize import word_tokenize
var2 = word_tokenize(text)
print(var2)
```

```
['Gayatri', 'is', 'sitting', 'beside', 'me', ',', 'doing', 'practical', 'A7', 'of', 'DSBDA', '.', 'Harish', 'is', 'sitting', 'far', 'awa
```

```
from nltk.corpus import stopwords
var3 = set(stopwords.words('english'))
print(var3)
```

```
{'hasn', 'isn', 'the', 'where', 't', 'weren', 'o', "hadn't", 'same', 'they', "it's", 'to', 'himself', 'if', "shouldn't", 'but', 'of', 'y
```

```
text = re.sub('[^a-zA-Z]',' ',text)
print(text)
```

```
Gayatri is sitting beside me  doing practical A  of DSBDA  Harish is sitting far away which is very sad
```

special characters and numbers gone

```
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
  if w not in var3:
    filtered_text.append(w)
print("TOkenized Sentence:",tokens)
print("Filtered Sentence:",filtered_text)
```

```
TOkenized Sentence: ['gayatri', 'is', 'sitting', 'beside', 'me', 'doing', 'practical', 'a', 'of', 'dsbda', 'harish', 'is', 'sitting', 'f
Filtered Sentence: ['gayatri', 'sitting', 'beside', 'practical', 'dsbda', 'harish', 'sitting', 'far', 'away', 'sad']
```

```
tokens = word_tokenize(text.upper())
filtered_text=[]
for w in tokens:
  if w not in var3:
    filtered_text.append(w)
print("TOkenized Sentence:",tokens)
print("Filtered Sentence:",filtered_text)
```

```
TOkenized Sentence: ['GAYATRI', 'IS', 'SITTING', 'BESIDE', 'ME', 'DOING', 'PRACTICAL', 'A', 'OF', 'DSBDA', 'HARISH', 'IS', 'SITTING', 'F
Filtered Sentence: ['GAYATRI', 'IS', 'SITTING', 'BESIDE', 'ME', 'DOING', 'PRACTICAL', 'A', 'OF', 'DSBDA', 'HARISH', 'IS', 'SITTING', 'FA
```

```
from nltk.stem import PorterStemmer
var4 = [ 'read','reading','reads','writing', 'written', 'wrote','writes','write',]
ps = PorterStemmer()
for w in var4:
  rootWord = ps.stem(w)
print(rootWord)
```

```
write
```

performing Lemmatization

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tt = nltk.word_tokenize(text)
print(text)
for w in tt:
  print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
studies studying cries cry
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

```
#term_frequency = no. of that words in doc/ total num of words in doc
import pandas as pd
import math
from sklearn.feature_extraction.text import TfidfVectorizer
doc_a = "We are executing dsbda practical of text analytics and it is 10.41 in the morning. Today is friyay!"
doc_b = "this is something? maybe another sentence. okay 123."
bowa = doc_a.split(" ")
bowb = doc_b.split()
print(bowa)
print(bowb)
```

```
['We', 'are', 'executing', 'dsbda', 'practical', 'of', 'text', 'analytics', 'and', 'it', 'is', '10.41', 'in', 'the', 'morning.', 'Today'
['this', 'is', 'something?', 'maybe', 'another', 'sentence.', 'okay', '123.']
```

```
uniqueWords = set(bowa).union(set(bowb))
print(uniqueWords)
```

```
{'is', 'analytics', '10.41', 'We', 'dsbda', 'maybe', 'something?', 'are', 'okay', 'and', 'Today', 'the', 'in', 'of', 'text', 'practical'
```

```
numOfWordsInA = dict.fromkeys(uniqueWords, 0)
print(numOfWordsInA)
```

```
{'is': 0, 'analytics': 0, '10.41': 0, 'We': 0, 'dsbda': 0, 'maybe': 0, 'something?': 0, 'are': 0, 'okay': 0, 'and': 0, 'Today': 0, 'the'
```

```
for word in bowa:
  numOfWordsInA[word] += 1
print(numOfWordsInA)
```

```
{'is': 2, 'analytics': 1, '10.41': 1, 'We': 1, 'dsbda': 1, 'maybe': 0, 'something?': 0, 'are': 1, 'okay': 0, 'and': 1, 'Today': 1, 'the'
```

```
numOfWordsInB = dict.fromkeys(uni, 0)
for word in bowb:
  numOfWordsInB[word] += 1
print(numOfWordsInB)
```

```
{'is': 1, 'analytics': 0, '10.41': 0, 'We': 0, 'dsbda': 0, 'maybe': 1, 'something?': 1, 'are': 0, 'okay': 1, 'and': 0, 'Today': 0, 'the'
```

```
def computeTF(wordDict, bagOfWords):
  tfDict={}
  bagOfWordsCount = len(bagOfWords)
  for word,count in wordDict.items():
    tfDict[word] = count/float(bagOfWordsCount)
  return tfDict
tfA = computeTF(numOfWordsInA, bowa)
tfB = computeTF(numOfWordsInB, bowb)
print(tfA)
print(tfB)
```

```
{'is': 0.1111111111111111, 'analytics': 0.05555555555555555, '10.41': 0.05555555555555555, 'We': 0.05555555555555555, 'dsbda': 0.0555555
{'is': 0.125, 'analytics': 0.0, '10.41': 0.0, 'We': 0.0, 'dsbda': 0.0, 'maybe': 0.125, 'something?': 0.125, 'are': 0.0, 'okay': 0.125, '
```

```
def computeIDF(documents):
  N = len(documents)
  idfDict = dict.fromkeys(documents[0].keys(), 0)
  for document in documents:
    for word, val in document.items():
      if val > 0:
        idfDict[word] += 1
  for word, val in idfDict.items():
    idfDict[word] = math.log(N/float(val))
  return idfDict
idfs = computeIDF([numOfWordsInA, numOfWordsInB])
print(idfs)
```

```
{'is': 0.0, 'analytics': 0.6931471805599453, '10.41': 0.6931471805599453, 'We': 0.6931471805599453, 'dsbda': 0.6931471805599453, 'maybe'
```

```
idfs
```

```
{'is': 0.0,
 'analytics': 0.6931471805599453,
 '10.41': 0.6931471805599453,
 'We': 0.6931471805599453,
 'dsbda': 0.6931471805599453,
 'maybe': 0.6931471805599453,
 'something?': 0.6931471805599453,
 'are': 0.6931471805599453,
 'okay': 0.6931471805599453,
 'and': 0.6931471805599453,
 'Today': 0.6931471805599453,
 'the': 0.6931471805599453,
 'in': 0.6931471805599453,
 'of': 0.6931471805599453,
 'text': 0.6931471805599453,
 'practical': 0.6931471805599453,
 'friyay!': 0.6931471805599453,
 'sentence.': 0.6931471805599453,
 'morning.': 0.6931471805599453,
 'executing': 0.6931471805599453,
 'it': 0.6931471805599453,
 'another': 0.6931471805599453,
 'this': 0.6931471805599453,
 '123.': 0.6931471805599453}
```

```
Start coding or generate with AI.
```